INTERN CODING CHALLENGE

The Challenge
The challenge is to write a function that accepts an arbitrarily-deep nested Array-like structure and returns a flattened structure with any null values removed. Include some documentation about how your solution works, including any cases in which it might fail. If time permits, please include a few test cases. You may implement the challenge using whatever programming language you're most comfortable with.

LANGUAGE – JAVA

APPROACH
I am using the generic type in Java, initially passing the collection type, which is an array-like structure, it includes ArrayList, List, LinkedList and various other structures. Later when I am executing the function. I am checking whether the passed argument is null or not.

a. **If null**, executing the else condition and printing "Collection is null".
b. **If not null**, Using for loop for each element. Every element can be an another collection of object type or any type of data type.

    a. If it is an instance of collection, call the flatten function (recursion).
    b. Else, check whether the object type (that would be datatype) is null or not, if not null store it in an ArrayList (array structure) of object type, which can store any data type.

c. Return the resultant ArrayList.

Below is the step by step explanation of my code:

```
/* Declaring and initializing an ArrayList (array like data structure) to
   store the resultant array. ArrayList can store any data type. It is of object type */
public static ArrayList<Object> result = new ArrayList<>();

/*
   flatten function whose return type is array like structure
   and taking the Generic Collection as input type.

   It can be any Array like structure (example List, LinkedList, HashSet)
   structure and can be deep nested. This fulfill are requirement for
   deep-nested array like structure.

   <T> is used for defining the Generic datatype, It would be of object type,
   which can be primitive data type or other collection
   Collection is used for Generic Array like structure.
*/
public static <T> ArrayList<Object> flatten(Collection<T> input) {
```

INTERN CODING CHALLENGE

```
/* This if condition checks, if the passed argument is null or not.
   If the collection is null, the function will not
   enter the if loop and just pass the message,
   collection is null
*/
if(input != null) {
    /*
    Once we determine the collection is not null.
    We are going iterate through each element using
    for loop. Each element can be another collection
    (array-like structure) or primitive data type.
    */
    for(T type : input) {

     // this if condition checks whether the object is instance of collection
      if(type instanceof Collection) {
        /*
            If the object is of type Collection,
            same flatten function is going to be
            called using recurssion
         */
        flatten((Collection)type);
      }
      /* this condition checks whether the object is anything
        other than collection, i.e., it is a primitve datatype */
      else {

        /* If the data type is  not null,
            value is going to be save in the resultant arrayList */
        if(type != null)
            result.add(type);
      }
    }
}
// Else loop will execute, if the collection is null
else
    System.out.println("Passed collection is null");

/*
   Returning the resultant arrayList(flattened array) whose
   has all the values except null. It is not deep nested array
*/
return result;
}
```

INTERN CODING CHALLENGE

Test Cases:

1. `{0, 2, {{2, 3}, 8, 100, null, {{null}}}, -2]};`
2. `null`
3. `{}`
4. `{{{{}}}}`
5. `{{null, null}, {3, -10, 4.7}, {}}`
6. `Large Values of 10 digits`